Computer Science CS134 (Spring 2020)
Shikha Singh & Iris Howley
Laboratory 3
Building a Python Toolbox (due Wednesday/Thursday at 11pm)

Objective. To construct a toolbox of functions for manipulating words.

This week we'll construct a small module of tools for manipulating words from word lists. When finished, we'll be able to answer some trivia questions.

Motivation: The NPR Puzzle. Will Shortz is the editor of the New York Times Crossword and the Puzzlemaster at National Public Radio. Each Sunday morning he challenges listeners with a puzzle to solve by the following Thursday. Typically these are language-based challenges, but, as we'll see, their solutions can frequently be computed.
Here are some interesting problems:

p1: (Proposed February 11, 2018.) Name part of the human body in six letters. Add an 'r' and rearrange the result to name a part of the body in seven letters. What is it?

p2: (Proposed September 24, 2017.) Think of a familiar 6-letter boy's name starting with a vowel. Change the first letter to a consonant to get another familiar boy's name. Then change the first letter to another consonant to get another familiar boy's name. What names are these?

p3: (Proposed April 2, 2017 by David Edelheit of Oyster Bay, N.Y.) Think of four 4-letter proper names that are all anagrams of each other. Two of them are first names–one male and one female. The other two are well-known geographical names. What names are these?

p4: (Proposed September 23, 2018 by Jim Levering of San Antonio) Think of an affliction in five letters. Shift each letter three spaces later in the alphabet—for example, 'a' would become 'd', 'b' would become 'e', etc. The result will be a prominent name in the Bible. Who is it?

Are you up for solving one or more of these challenges?

Getting Started. Before you begin, clone the starter repository for this week's lab into your `cs134` directory as you did last week, by going to `https://evolene.cs.williams.edu` and clicking `Clone` and the the `Copy URL to clipboard` button under the `Clone with HTTPS` text. Return to the `Terminal`, navigate to your cs134 directory (with `cd ~/cs134`, if you are on a lab machine), and type `git clone` followed by the URL of the lab project you just copied, followed by the name of the lab `lab03`. Your line should look something like this:

```
git clone https://evolene.cs.williams.edu/cs134-s20/lab03/22xyz3.git lab03/
```

where your CS username replaces 22xyz3.

This week's tasks.

1. Build a small toolbox of string-related functions, `wordTools.py`:

   ◇ Write a function, `words(wlfile)`, that returns a `list` of words found one per line (one word may include spaces) in a file whose name is specified by `wlfile`. Strip off any unnecessary whitespace from the words as you read them in. You might use it this way:

   ```
   >>> len(words('wordlists/firstNames'))
   5166
   >>> words('wordlists/bodyParts')[124]
   'skeleton'
   ```

   ◇ Write a function, `sized(n,l)`, that takes a word list `l` and a word length, `n`. It returns the words in the list that are exactly length `n`. For example:

   ```
   >>> sized(6,words('wordlists/firstNames'))[0:3]
   ['Adelia', 'Adella', 'Adelle']
   ```

   ◇ Write a function, `canon(s)`, that returns a lower-cased, space-free, and sorted rearrangement of the letters of `s`. For example:

   ```
   >>> canon('Lot A')
   'alot'
   >>> canon('aim')
   'aim'
   >>> canon('Mia')
   'aim'
   ```

   You may develop other useful functions, as well. If you do, collect them in `wordTools.py`.

2. Make sure your `wordTools` toolkit is a solidly built module:

   (a) There is a triple-quoted docstring that appears at the top of the file that helps the user understand the purpose of this module. You can check this out with: `pydoc3 wordTools`

   (b) It defines `__all__` to be a list of strings of the names that should be imported when you write:

   ```
   from wordTools import *
   ```

   (c) Make sure that every function is documented with a docstring.

   (d) Thoroughly test each of the functions. Consider what the output should be for:

   - `canon(..)`: `'LowerCase'`, `'s p a c e s'`, and `'sorted'`
   - `len(words('wordlists/bibleNames'))`: 2114, `len(words('wordlists/bibleNames')[0])`: 7
   - `sized(0,[])`: `[]`, `len(sized(6,words('wordlists/bibleNames')))`: 517

   (e) Include at least two doctests (>>>) for each function in `wordTools`

3. Now, solve at least one of the puzzles. For example: to solve puzzle 3, write a new python script, `p3.py` that prints either the solution directly or no more than 10 possible solutions to be considered. The word lists found in the `wordlists` folder will be useful.

Good luck! Do not forget to stage, commit, and push your work as it progresses!

Submitting your work. When you're finished, stage, commit, and push your work to the server as you did in Lab 1 & 2. Remember that you must certify that your work is your own, by typing out the Honor Code statement in the `honorcode.txt` file, committing and pushing it along with your work.

Late days. You are allowed a total of 3 late days over the semester, with at most 2 late days towards any one lab. You must request a late day in advance on the form, here: `http://bit.ly/s20late`.

Working across different machines. If you clone the lab repository on two different machines and are planning to switch between them, you must always start by "pulling" your most recent work from the server to the local machine. You can do this by adding the correct project folder to Atom, toggling the `Git` tab, and clicking on the `Fetch` button at the bottom right of the `Git` tab, followed by `Pull`.

Grading Guidelines. Both functionality and programming style are important when writing code, just as both the content and the writing style are important when writing an essay. In this program, some of the specific functional requirements we will test for include:

- Your code for the problems must generate the answers to the puzzle you have chosen. That is, `p3.py` must use code to **find** the potential answers, it cannot simply print the correct answers.
- The most efficient solutions are those with the fewest number of (nested) loops. `p1` can be best solved using 2 loops. `p2` and `p3` can be solved with 2-3 loops. It is possible to solve `p4` using 2 loops. Can you find an even better solution?!
- Just like last week's lab, we require that the functions defined in `wordTools.py` follow our specifications, e.g., `sized` takes in two parameters—a number `n` and a list `l`—in that order and returns a list. Do not modify the function names, their parameters, nor what is returned. The examples above in **This week's tasks** provide details about these constraints.
- It is very important that your files be named properly so that we can run our tests for grading. The only acceptable names and capitalization are: `wordTools.py`, `p1.py`, `p2.py`, `p3.py`, and `p4.py`.

Stylistically, we expect to see programs that exhibit the following: meaningful names used in declarations, informative comments (both in-line and docstrings), good and consistent formatting, and good choice of Python commands. There is some subjectivity to what makes good style, but the basic goal is to make your ideas as clear and easy to follow as possible.

Grading scale. Programming labs will be graded on the following scale:

A+	An absolutely fantastic submission of the sort that will only come along a few times during the semester.
A	A submission that exceeds our standard expectations for the assignment. The program must reflect additional work beyond the requirements or get the job done in a particularly elegant way.
A-	A submission that satisfies all the requirements for the assignment – a job well done.
B+	Submission meets the requirements for the assignment, possibly with a few small problems.
B	A submission that has problems serious enough to fall short of the requirements for the assignment.
C	A submission that has extremely serious problems, but nonetheless shows some effort & understanding.
D	A submission that shows little effort and does not represent passing work.

Extra credit—expanding your knowledge. We're only requiring you solve one problem this week to get credit, but if you'd like to dig deeper feel free to solve multiple word problems!

⋆